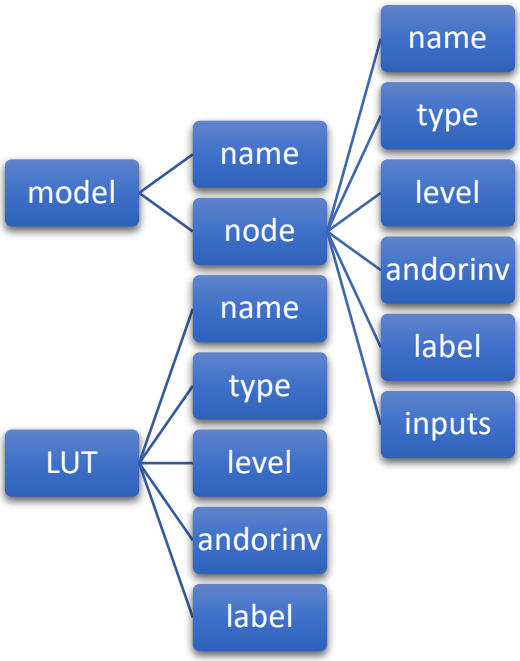
CS613200 FINAL PROJECTFINAL PROJECT

資工所 賴御誠 109062701

II. DATA STRUCTURE



The main structure is model, which handles the input file, and the entire nodes.

Nodes are gates that consists of all parameters that are related to gate, the adjacent nodes are located at inputs.

LUT is the output data structure after mapping is done, the internal structure is same as nodes.

III. ALGORITHM

The algorithm that the program uses is from the paper “DAG-Map: Graph Based FPGA Technology Mapping For Delay Optimization”, which is near optimal for mapping K-LUT FPGA.

Meanwhile, the algorithm used in “FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs” is also implemented at first but canceled due to the difficulty of tracking cuts and using augmenting path algorithm. These two papers proposed a similar workflow. Therefore, most of the codebase are shared. (before finding K feasible cuts)

The algorithm consists of two main stages: Decomposition and DAG Map.

In decomposition stage, the multi-input gate will be transformed into two input gate using the following flow:

algorithm decompose-multi-input-gate (DMIG)
  let $V = input(v) = \{u_1, u_2, ..., u_m\}$;
  while $|V| > 2$ **do**
    let u_i and u_j be the two nodes of V with smallest levels;
    introduce a new node x;
    $input(x) = \{u_i, u_j\}$;
    $level'(x) = \max(level'(u_i), level'(u_j)) + 1$;
    $V = (V - \{u_i, u_j\}) \cup \{x\}$
  end-while;
  Connect the only two nodes left in V to v as its inputs;
  Return the binary tree $T(v)$ rooted at v;
end-algorithm.

Fig. 2 Algorithm DMIG.

After decomposition, the circuit will be left with only two input gates.

Input: 10aoi_sample01.blif

Output:

.model sample01

.inputs a b c d e

```
.outputs f g

.names d n6621 n662

11 1

.names d r6621 r662

1- 1

-1 1

.names a n662 f

1- 1

-1 1

.names n662 t6621 t662

11 1

.names t662 g

0 1

.names c b n6621

11 1

.names c b r6621

1- 1

-1 1

.names e r662 t6621

11 1

.end
```

Next, DAG Map stage will be done to map the circuit to K-LUT FPGA, the algorithm can separate into two phases: labeling phase and mapping phase.

During labeling phase, the circuit will be traversed to find the level of the final FPGA decomposition. Then, mapping phase will use the circuit labeling and the K value to map the circuit into multiple K-LUT gates. The flow is as follows:

algorithm DAG-Map
  /* step 1: labeling the network */
  for each PI node v **do**
    $h(v) = 0$;
  T = list of non-PI nodes in topological order;
  while T is not empty **do**
    remove the first node v from T;
    let $p = \max\{h(u) \mid u \in input(v)\}$;
    if $|input(N_p(v) \cup \{v\})| \leq K$
    then $h(v) = p$
    else $h(v) = p + 1$
  end-while;
  /* step 2 : generate K-LUTs */
  L = list of PO nodes;
  while L contains non-PI nodes **do**
    remove a non-PI node v from L, i.e. $L = L - \{v\}$;
    introduce a K-LUT v' to implement the function of v such that
      $input(v') = input(N_{h(v)}(v))$;
    $L = L \cup input(v')$
  end-while;
end-algorithm.

Fig. 4 Algorithm DAG-Map.

Input: 10aoi_sample01.blif (After Stage 1)

Output: With K=4

.model sample01

.inputs a b c d e

.outputs f g

.names t662 g

0 1

.names n662 t6621 e r662 t662

```
1111 1

.names d r6621 c b r662

1--- 1

-1-- 1

--1- 1

---1 1

.names c b r6621

1- 1

-1 1

.names e r662 t6621

11 1

.names d n6621 c b n662

1111 1

.names c b n6621

11 1

.names a n662 f

1- 1

-1 1

.end
```

The complexity of the code is about O(N^3k), this is due to multiple loops for searching and updating the data structure. Improving the structure and breakpoints could reduce the complexity.

IIII. RESULTS

Here are the results of the runtime:

```
[yclai20@ic55 ~/ALS]$ make
g++ main.cpp -o map
[yclai20@ic55 ~/ALS]$ ./map -k 4 10aoi_sample01.blif output.blif
The circuit level is 2.
The number of LUTs is 8.
[yclai20@ic55 ~/ALS]$ ./map -k 2 10aoi_sample01.blif output.blif
The circuit level is 4.
The number of LUTs is 8.
[yclai20@ic55 ~/ALS]$ make clean
rm -f map
[yclai20@ic55 ~/ALS]$
[yclai20@ic55 ~/ALS]$
```

```
[yclai20@ic55 abc]$ ./abc
UC Berkeley, ABC 1.01 (compiled Apr 12 2021 10:46:01)
abc 01> cec 10aoi_sample01.blif output.blif
Networks are equivalent after structural hashing.  Time =     0.00 sec
abc 01>
```